Artificial Intelligence 61 (1993) 291–306
Elsevier

ARTINT 1055

Research Note

An automatic proof of Gödel's incompleteness theorem *

Kurt Ammon

Windmühlenweg 27, W-2000 Hamburg 52, Germany

Received June 1991
Revised July 1992

Abstract

Ammon, K., An automatic proof of Gödel's incompleteness theorem (Research Note), Artificial Intelligence 61 (1993) 291–306.

The SHUNYATA program contains heuristics which are related to reasoning processes of mathematicians and guide the search for a proof. For example, a heuristic applies the method of reductio ad absurdum to prove the negation of a proposition. Another heuristic generates formulas and sets which form the central "ideas" of significant proofs. Some heuristics control the application of other heuristics, for example, by time limits which interrupt a heuristic if it achieves no result. The architecture and the mode of operation of SHUNYATA are illustrated in detail by SHUNYATA's proof of Gödel's incompleteness theorem which says that every formal number theory contains an undecidable formula, i.e., neither the formula nor its negation are provable in the theory. In this proof, SHUNYATA constructed several closed formulas on the basis of elementary rules for the formation of formulas and proved that one of these formulas is undecidable. Further experiments with a learning procedure suggest that an automatic construction of SHUNYATA's heuristics on the basis of a universal set of elementary functions is feasible.

Correspondence to: K. Ammon, Windmühlenweg 27, W-2000 Hamburg 52, Germany.

*This work, in whole or in part, describes components of machines or processes protected by one or more patents or patent applications in Europe, the United States of America, or elsewhere. Further information is available from the author.

0004-3702/93/$ 06.00 © 1993 — Elsevier Science Publishers B.V. All rights reserved

1. Introduction

The main problem in the proof of Gödel's theorem is the construction of an undecidable formula. In order to construct such a formula, a mathematician may apply elementary rules for the formation of formulas to a known formula and attempt to prove that one of the resulting formulas is undecidable. An example of a rule for the formation of formulas is that $\neg F$ is a formula if F is a formula. Another rule is that $\forall x F$ is a formula if x is a variable and F a formula. The application of the two rules to the variable x and the formula A(x) yields the formulas $\neg$A(x) and $\forall$xA(x). The SHUNYATA program contains heuristics which are related to reasoning processes of mathematicians, for example, a heuristic which produces an undecidable formula on the basis of elementary rules for the formation of formulas. In preparatory experiments, SHUNYATA produced proofs of Banach's fixed point theorem (Ammon [1]) and the Schröder–Bernstein theorem.[1] In more recent experiments, it generated proofs of the Bolzano–Weierstrass theorem and the theorem that every continuous function on a compact set is uniformly continuous (see Bledsoe [8, p. 31]).[2] SHUNYATA's architecture and mode of operation are illustrated by its proof of Gödel's incompleteness theorem.[3] Section 2 introduces a language in which the theorem and the proof are represented. Section 3 describes heuristics which were applied to generate a proof of the theorem. Sections 4 and 5 give Gödel's theorem and an automatic proof which was generated by the heuristics.

2. Language

Gödel's theorem and its proof are represented in the language CL which incorporates concepts such as the concept of substitution into special reasoning procedures and which models the ordinary representation of proofs in mathematics textbooks.

The ordinary notation of the result of substituting a term t for the variable x in a formula $F(x)$ is $F(t)$ (see [11, p. 78]). In the language CL, the formula $F(t)$ is represented by the expression $sub(F, x, t)$. The commutativity of substitution with propositional connectives and quantifiers is incorporated into special reasoning procedures. For example, the commutativity of substitution with negation says that the formula $sub(\neg F, x, t)$ is equivalent to the formula $\neg sub(F, x, t)$. The commutativity of substitution

[1] Kaufmann [10] checked a proof of the Schröder–Bernstein theorem.
[2] Ballantyne and Bledsoe [7] generated proofs of these theorems in nonstandard analysis.
[3] Shankar [13] checked a detailed proof of Gödel's incompleteness theorem.

with universal quantifiers says that the formula $sub(\forall y F, x, t)$ is equivalent to the formula $\forall y\, sub(F, x, t)$ if the variable x is different from the variable y. The commutativity of substitution with propositional connectives and quantifiers is correct because it is contained in the ordinary definition of substitution (see [11, p. 78]). Formulas are normalized such that the formula F in the first argument of an expression $sub(F, x, t)$ does not contain propositional connectives or quantifiers. Section 6 discusses why formulas are normalized in this way. In order to achieve a complete encoding of substitution, normalized formulas are translated into equivalent formulas if required. An example: Let t be a term which is free for the variable x in a formula $F(x)$. The rule of existence introduction implies that $\exists x F(x)$ is provable if $F(t)$ is provable (see [11, p. 99]). A formalization in the language CL is

$$all\, F, x, t(provable(sub(F, x, t)) \rightarrow provable(\exists x F)). \tag{1}$$

Let A(x) be a formula such that $\neg$A(**n**) is provable for a numeral **n** which represents the natural number n in formal number theory (see Kleene [11, p. 195]). An immediate application of the rule of existence introduction (1) to the provable formula $\neg$A(**n**), which is represented by the normalized expression

$$\neg sub(\mathrm{A(x)}, \mathrm{x}, \mathbf{n}), \tag{2}$$

is impossible because the expression $sub(F, x, t)$ in the antecedent of (1) cannot be matched against (2). Therefore, the expression (2) is translated into the equivalent expression

$$sub(\neg \mathrm{A(x)}, \mathrm{x}, \mathbf{n}) \tag{3}$$

according to the commutativity of substitution with negation. The matching of the expression $sub(F, x, t)$ in (1) against (3) yields the substitution $\neg$A(x)/F, x/x, and **n**/t and thus the instantiated consequent $provable(\exists \mathrm{x} \neg \mathrm{A(x)})$ of (1). Furthermore, normalized formulas containing two or more consecutive applications of substitution are also translated into equivalent formulas if required. An example: let A(x, y) be a formula such that A(**k**, **n**) is provable for some numerals **k** and **n**. The formula A(**k**, **n**) can be represented by the expression

$$sub(sub(\mathrm{A(x, y)}, \mathrm{x}, \mathbf{k}), \mathrm{y}, \mathbf{n}). \tag{4}$$

The matching of the expression $sub(F, x, t)$ in (1) against (4) yields the substitution $sub(\mathrm{A(x, y)}, \mathrm{x}, \mathbf{k})/F$, x/$x$, and **n**/$t$ and thus the instantiated consequent

$$provable(\exists \mathrm{x}\, sub(\mathrm{A(x, y)}, \mathrm{y}, \mathbf{n})) \tag{5}$$

of (1), i.e., *provable*($\exists$xA(x, **n**)) in ordinary notation. The expression (4) is also translated into the equivalent expression

$$sub(sub(\mathrm{A}(\mathrm{x},\mathrm{y}),\mathrm{y},\mathbf{n}),\mathrm{x},\mathbf{k}). \tag{6}$$

The matching of the expression $sub(F,x,t)$ in (1) against (6) yields the substitution $sub(\mathrm{A}(\mathrm{x},\mathrm{y}),\mathrm{y},\mathbf{n})/F$, y/x, and $\mathbf{k}/t$ and thus another consequent *provable*($\exists$yA(**k**, y)) of (1).

3. Heuristics

The SHUNYATA program contains heuristics which are related to reasoning processes of mathematicians and guide the search for a proof. We describe the heuristics required for a proof of Gödel's theorem.

The input of a heuristic is a *goal*, which may contain a constraint such as a time limit, and a *partial proof*, which is a list of proof steps. A *proof step* contains a proposition or an assumption, and an explanation which gives the heuristic, the preceding proof steps, definitions, and lemmas that were used to derive the proposition.

The heuristics apply derived rules of inference of Kleene's formal system [11, pp. 69–213]. These rules describe logical operations as introduction and elimination of logical symbols [11, pp. 98–102]. They are more or less similar to the rules of inference of Gentzen's system of natural deduction [9]. The heuristics can be regarded as control procedures which guide the selection and instantiation of the rules of inference. SHUNYATA's architecture includes a complete proof procedure which can be executed simultaneously with the heuristics. This is achieved by means of time limits which assign a low priority to the proof procedure. Thus, the heuristics change the order of the selection and instantiation of the rules of inference without affecting SHUNYATA's completeness.

The *proof* heuristic attempts to prove a proposition. If the proposition is not contained in the proof, this heuristic finds out the type of the proposition such as a universal proposition, an implication, an existential proposition, or a negation, and calls special heuristics that process propositions of a special type. Otherwise, the proof heuristic calls the reduction heuristic.

The *universal* heuristic proves universal propositions $\forall x(P(x) \rightarrow Q(x))$. It introduces the assumption that $P(x)$ holds for x and calls the proof heuristic to prove $Q(x)$. Then, it applies the rules of implication and generality introduction [11, pp. 98–99], which yields the universal proposition to be proved.

The *implication* heuristic proves implications $A \rightarrow B$. It introduces the assumption that A holds and calls the proof heuristic to prove B. Then,

it applies the rule of implication introduction [11, p. 98] which yields the implication to be proved.

The *existential* heuristic proves existential propositions $\exists x P(x)$. If $P(x)$ says that x is a formula or a predicate, the existential heuristic calls the composition heuristic which constructs examples of such formulas or predicates. Then, it calls the proof heuristic to prove that $P(x)$ holds for one of the formulas or predicates. It controls the proof heuristic by a time limit which is one minute at the beginning and which is doubled if a proof is not achieved. The application of the rule of existence introduction [11, p. 99] to a formula or predicate x with $P(x)$ yields the existential proposition $\exists x P(x)$ to be proved.

The *composition* heuristic constructs composed formulas and predicates by applying elementary rules for the formation of formulas and predicates to the partial proof. Roughly speaking, the rules give the domains and ranges of elementary functions which form the elementary building stones of the formulas and predicates. The application of the rules yields compositions of the elementary functions. Some of these compositions are formulas or predicates. In the proof of Gödel's theorem, the composition heuristic constructs an undecidable formula. It applies seven elementary rules for the formation of formulas. The first three rules say that x and y are variables of formal number theory and that $\mathrm{A(x,y)}$ is a formula with the variables x and y. Two further rules say that $\neg F$ and $\forall x F$ are formulas if F is a formula and x is a variable [11, p. 72]. Another rule says that the Gödel number of a formula is a natural number. The last rule says that $F(\mathbf{n})$ is a formula if $F(x)$ is a formula with the variable x and n is a natural number. The formula $F(\mathbf{n})$ is obtained by substituting the numeral $\mathbf{n}$, which represents the natural number n in the formal theory [11, p. 195] for the variable x in $F(x)$. For example, the repeated application of these rules yields that $\neg\mathrm{A(x,y)}$, $\forall\mathrm{xA(x,y)}$, $\forall\mathrm{yA(x,y)}$, $\forall\mathrm{x}\neg\mathrm{A(x,y)}$, and $\forall\mathrm{y}\neg\mathrm{A(x,y)}$ are formulas. If the application of a rule yields a set of closed formulas, these formulas form the output of the composition heuristic. If this set contains short and long formulas, subsequent heuristics such as the existential heuristic first process the short formulas. The composition heuristic also produces sets by constructing predicates that determine the elements of these sets.

The *negation* heuristic proves the negation of a proposition. It first calls the reduction heuristic to prove the negation within a time limit. If this fails, it calls the contradiction heuristic.

The *reduction* heuristic uses a universal proposition $\forall x(P(x) \rightarrow Q(x))$ to reduce the goal to prove a proposition $Q(a)$ to the subgoal to prove $P(a)$ as follows: It matches the consequent $Q(x)$ of the universal proposition against the proposition $Q(a)$ to be proved and generates the subgoal to prove the instance $P(a)$ of its antecedent $P(x)$. Then, it uses the rules of generality and implication elimination [11, pp. 98–99] to apply the universal

proposition to $P(a)$ which yields the proposition $Q(a)$ to be proved.

The *contradiction* heuristic applies the method of reductio ad absurdum to prove the negation of a proposition. First, it applies definitions and lemmas to the proof. Then, it generates the assumption that the proposition holds and calls the forward heuristic. If the forward heuristic discovers a contradiction, the contradiction heuristic applies the rule of reductio ad absurdum [11, p. 99], which yields the negation to be proved.

The *forward* heuristic repeatedly applies universal propositions to the proof and selects the proof steps whose explanation refers to an immediately preceding proof step. It uses the rules of generality and implication elimination [11, pp. 98–99] to apply a universal proposition $\forall x(P(x) \rightarrow Q(x))$ to a proposition $P(a)$ which yields the proposition $Q(a)$. If $P(x)$ has the form $\exists y R(x,y)$, it first applies the rule of existence introduction to a proposition $R(a,b)$ which yields $\exists y R(a,y)$, i.e., $P(a)$. If $Q(x)$ has the form $\exists y R(x,y)$, it additionally applies the rule of existence elimination to $Q(a)$, i.e., $\exists y R(a,y)$, which yields that $R(a,b)$ holds for some b.

The SHUNYATA program contains more sophisticated variants of the heuristics described above. For example, the reduction and the forward heuristic process more complex universal propositions such as $\forall x(P(x) \leftrightarrow Q(x))$ which are translated into universal propositions containing a single implication such as $\forall x(P(x) \rightarrow Q(x))$ and $\forall x(Q(x) \rightarrow P(x))$. The translated propositions are used as described above. Furthermore, the heuristics process propositions containing several variables.

4. Gödel's theorem

In the formalization of Gödel's incompleteness theorem and its proof in this and the following section, $fnt(T)$ means that T is a formal number theory, $num(n)$ stands for the numeral $\mathbf{n}$ of a natural number n, $gn(F)$ stands for the Gödel number of a formula F, and $gnp(n,F,T)$ means that n is the Gödel number of a proof of a formula F in a formal theory T (see [11, p. 206]). Gödel's theorem requires two definitions.

Definition 1. A formal theory is *consistent* if for no formula F both F and $\neg F$ are provable in the theory [11, p. 124]. A formalization is:

$$all\,T(fnt(T) \rightarrow (consistent(T) \leftrightarrow all\,F(provable(F,T) \rightarrow not(provable(\neg F,T))))).$$

Definition 2. A formal theory is *ω-consistent* if every formula $F(x)$ has the property: If $\vdash F(\mathbf{n})$, i.e., $F(\mathbf{n})$ is provable, for all natural numbers n, then

$\overline{\vdash \neg\forall x F(x)}$, i.e., $\neg\forall x F(x)$ is not provable (see Kleene [11, p. 207]). A formalization is:

$$\begin{aligned}
&\mathit{all}\, T(\mathit{fnt}(T) \rightarrow \\
&\quad (\omega\text{-}\mathit{consistent}(T) \leftrightarrow \\
&\qquad \mathit{all}\, F, x\, (\mathit{all}\, n\, (\mathit{number}(n) \rightarrow \\
&\qquad\qquad \mathit{provable}(\mathit{sub}(F, x, \mathit{num}(n)), T)) \rightarrow \\
&\qquad\quad \mathit{not}(\mathit{provable}(\neg\forall x F, T))))).
\end{aligned}$$

The SHUNYATA program generated a proof of Gödel's theorem in the form: There is a closed formula F of formal number theory such that

- if the theory is consistent, $\overline{\vdash F}$, i.e., F is not provable, and
- if the theory is ω-consistent, $\overline{\vdash \neg F}$, i.e., $\neg F$ is not provable.[4]

A formalization is:

$$\begin{aligned}
&\mathit{all}\, T(\mathit{fnt}(T) \rightarrow \\
&\quad \mathit{exists}\, F\, (\mathit{closed\text{-}formula}(F, T)\ \& \\
&\qquad (\mathit{consistent}(T) \rightarrow \mathit{not}(\mathit{provable}(F, T)))\ \& \\
&\qquad (\omega\text{-}\mathit{consistent}(T) \rightarrow \mathit{not}(\mathit{provable}(\neg F, T))))).
\end{aligned}$$

In contrast to some other forms of Gödel's theorem, the form given above does not explicitly contain an undecidable formula (see [11, pp. 207–208]). Thus, SHUNYATA's additional task was to "discover" an undecidable formula which is the most essential part of other forms of Gödel's theorem.

5. Proof

The SHUNYATA program took some nine hours to generate a proof of Gödel's theorem. We give the proof in ordinary representation, explain its computational representation, and describe how it was generated by the heuristics in Section 3. The proof requires the definition of a predicate $A(k, n)$ and five lemmas which were given to SHUNYATA.

Definition 3. The predicate $A(k, n)$ is defined as follows: k is the Gödel number of a formula $F(\mathrm{x})$ and n is the Gödel number of a proof of $F(\mathbf{k})$, [11, p. 206]. A formalization is:

[4] Rosser showed that the assumption of ω-consistency can be weakened to consistency (see [11, p. 208]).

$$all\ T, F, n\,(fnt(T) \rightarrow\ (A(gn(F), n) \leftrightarrow gnp(n, sub(F, \mathrm{x}, num(gn(F))), T)).$$

Lemma 1. *A formal number theory contains a formula* $\mathrm{A(x,y)}$ *which expresses the predicate* $A(k,n)$ *for the Gödel numbering (see* [11, p. 207]*). This means that* $\vdash \mathrm{A}(\boldsymbol{k},\boldsymbol{n})$*, i.e.,* $\mathrm{A}(\boldsymbol{k},\boldsymbol{n})$ *is provable, if* $A(k,n)$ *is true and that* $\vdash \neg\mathrm{A}(\boldsymbol{k},\boldsymbol{n})$*, i.e.,* $\neg\mathrm{A}(\boldsymbol{k},\boldsymbol{n})$ *is provable, if* $A(k,n)$ *is false. A formalization is:*

$$all\ T, k, n\,(fnt(T) \rightarrow$$
$$(A(k,n) \rightarrow provable(\mathrm{A}(num(k), num(n)), T))\ \&$$
$$(not(A(k,n)) \rightarrow provable(\neg\mathrm{A}(num(k), num(n)), T))).$$

Lemma 2. *A formula is provable if and only if there is a natural number which is the Gödel number of a proof of the formula (see* [11, p. 206]*). A formalization is:*

$$all\ T, F\,(fnt(T) \rightarrow (provable(F,T) \leftrightarrow exists\ n\,(gnp(n,F,T)))).$$

Lemma 3. *An* ω*-consistent formal theory is consistent (see* [11, p. 207]*). A formalization is:*

$$all\ T\,(fnt(T)\ \&\ \omega\text{-}consistent(T) \rightarrow consistent(T)).$$

Lemma 4. *Let* t *be a term which is free for the variable* x *in a formula* $F(x)$*. The rule of existence introduction implies that* $\exists x F(x)$ *is provable if* $F(t)$ *is provable (see* [11, p. 99]*). A formalization is:*

$$all\ T, F, x, t\,(fnt(T)\ \&\ provable(sub(F,x,t)) \rightarrow$$
$$provable(\exists x F, T)).$$

Lemma 5. *Each of* $\exists$ *and* $\forall$ *can be expressed in terms of the other and* $\neg$ *(see Kleene* [11, p. 162]*). A formalization is:*

$$all\ T, F, x\,(fnt(T) \rightarrow$$
$$(provable(\exists x F, T) \leftrightarrow provable(\neg\forall x \neg F, T))).$$

Proof. The proof in ordinary representation is: Let k be the Gödel number of the formula $\mathrm{B(x)}$ defined by $\forall\mathrm{y}\neg\mathrm{A(x,y)}$. We show that the closed formula $\mathbf{B(k)}$ is undecidable. We assume that the theory is consistent and $\vdash \mathbf{B(k)}$, i.e., $\mathbf{B(k)}$ is provable, and derive a contradiction. Because $\mathbf{B(k)}$ is provable, there is a natural number n which is the Gödel number of a proof of $\mathbf{B(k)}$. Therefore, $A(k,n)$ according to Definition 3. Because

$\mathrm{A}(\mathrm{x},\mathrm{y})$ expresses $A(k,n)$, $\vdash \mathrm{A}(\mathbf{k},\mathbf{n})$, i.e., $\mathrm{A}(\mathbf{k},\mathbf{n})$ is provable. By the rule of existence introduction (Lemma 4), $\vdash \exists \mathrm{y}\mathrm{A}(\mathbf{k},\mathrm{y})$. This implies $\vdash \neg\forall \mathrm{y}\neg\mathrm{A}(\mathbf{k},\mathrm{y})$, i.e., $\vdash \neg\mathrm{B}(\mathbf{k})$, because $\exists$ can be expressed by $\forall$ and $\neg$ according to Lemma 5. This yields a contradiction because of the consistency of the theory and $\vdash \mathrm{B}(\mathbf{k})$. It follows that, if the theory is consistent, $\overline{\vdash \mathrm{B}(\mathbf{k})}$, i.e., $\mathrm{B}(\mathbf{k})$ is not provable. Now, we assume that the theory is ω-consistent and prove $\overline{\vdash \neg\mathrm{B}(\mathbf{k})}$, i.e., $\neg\mathrm{B}(\mathbf{k})$ is not provable. Because the theory is ω-consistent, it is consistent according to Lemma 3. Therefore, $\overline{\vdash \mathrm{B}(\mathbf{k})}$, i.e., $\mathrm{B}(\mathbf{k})$ is not provable. Let n be a natural number. Because $A(k,n)$ implies $\vdash \mathrm{B}(\mathbf{k})$ and thus a contradiction, $A(k,n)$ is false. Therefore, $\vdash \neg\mathrm{A}(\mathbf{k},\mathbf{n})$ because $\mathrm{A}(\mathrm{x},\mathrm{y})$ expresses $A(k,n)$. Thus, $\vdash \neg\mathrm{A}(\mathbf{k},\mathbf{n})$ for all natural numbers n. Because of the ω-consistency, $\overline{\vdash \neg\forall \mathrm{y}\neg\mathrm{A}(\mathbf{k},\mathrm{y})}$, i.e., $\overline{\vdash \neg\mathrm{B}(\mathbf{k})}$. It follows that, if the theory is ω-consistent, $\overline{\vdash \neg\mathrm{B}(\mathbf{k})}$, i.e., $\neg\mathrm{B}(\mathbf{k})$ is not provable. This completes the proof. □

Table 1 contains the proof generated by the SHUNYATA program. It has thirty proof steps. Each proof step consists of a number, a proposition or an assumption, and an explanation which gives the heuristic and the preceding proof steps, definitions, and lemmas that were used to derive the proposition. For example, the proposition in step 5 is produced by the contradiction heuristic by applying Definition 1 to step 4. The proposition in step 7 is produced by the forward heuristic by applying Lemma 2 to steps 1 and 6. The second step introduces the name k for the Gödel number of the formula $\forall \mathrm{y}\neg\mathrm{A}(\mathrm{x},\mathrm{y})$. The expression "*exists* $F(...)$" in step 30 stands for the existential proposition in step 29. In order to improve the readability of the proof in Table 1, the numerals $\mathbf{k}$ and $\mathbf{n}$ stand for $num(k)$ and $num(n)$. Furthermore, the ordinary notation $F(t)$ of the result of substituting a term t for the variable x in a formula $F(x)$ stands for expressions $sub(F,x,t)$. Steps 9, 11, 21, and 27 use Definitions 2 and 3 and Lemma 4 which contain an expression of the form $sub(F,x,t)$. In the language CL, formulas in preceding proof steps are also translated into such a representation so that the expressions of the form $sub(F,x,t)$ in definitions and lemmas can be matched against these formulas (see Section 2). For example, the formula $\forall \mathrm{y}\neg\mathrm{A}(\mathbf{k},\mathrm{y})$ in step 8, which is represented by the expression $\forall \mathrm{y}\neg sub(\mathrm{A}(\mathrm{x},\mathrm{y}),\mathrm{x},\mathbf{k})$, is translated into the expression $sub(\forall \mathrm{y}\neg\mathrm{A}(\mathrm{x},\mathrm{y}),\mathrm{x},\mathbf{k})$ so that the application of Definition 3 to steps 1 and 8 yields the substitution T/T, $\forall \mathrm{y}\neg\mathrm{A}(\mathrm{x},\mathrm{y})/F$, and n/n and thus step 9. In step 11, the substitution for the variables T, F, x, and t in Lemma 4 is T/T, $\mathrm{A}(\mathbf{k},\mathrm{y})/F$, y/x, and $\mathbf{n}/t$. In step 21, the substitution for the variables T, F, and n in Definition 3 is T/T, $\forall \mathrm{y}\neg\mathrm{A}(\mathrm{x},\mathrm{y})/F$, and n/n. In step 27, the substitution for the variables T, F, and x in Definition 2 is T/T, $\neg\mathrm{A}(\mathbf{k},\mathrm{y})/F$, and y/x.

Table 1
SHUNYATA's proof of Gödel's incompleteness theorem.

Step	Proposition or assumption	Explanation
1.	*assume* T (*fnt*(T))	Universal
2.	*name*(k, *gn*($\forall y \neg A(x, y)$))	
3.	*closed-formula*($\forall y \neg A(\mathbf{k}, y), T$)	Composition
4.	*assume*(*consistent*(T))	Implication
5.	*all* F (*provable*(F, T) $\rightarrow$ *not*(*provable*($\neg F, T$)))	Contradiction, D1, 4
6.	*assume*(*provable*($\forall y \neg A(\mathbf{k}, y), T$))	Contradiction
7.	*exists* n (*gnp*($n, \forall y \neg A(\mathbf{k}, y), T$))	Forward, L2, 1, 6
8.	*assume* n (*gnp*($n, \forall y \neg A(\mathbf{k}, y), T$))	Forward, 7
9.	$A(k, n)$	Forward, D3, 1, 8
10.	*provable*($A(\mathbf{k}, \mathbf{n}), T$)	Forward, L1, 1, 9
11.	*provable*($\exists y A(\mathbf{k}, y), T$)	Forward, L4, 1, 10
12.	*provable*($\neg \forall y \neg A(\mathbf{k}, y), T$)	Forward, L5, 1, 11
13.	*not*(*provable*($\neg \forall y \neg A(\mathbf{k}, y), T$))	Forward, 6, 5
14.	*not*(*provable*($\forall y \neg A(\mathbf{k}, y), T$))	Contradiction, 12, 13, 6
15.	*consistent*(T) $\rightarrow$ *not*(*provable*($\forall y \neg A(\mathbf{k}, y), T$))	Implication, 4, 14
16.	*assume*(ω-*consistent*(T))	Implication
17.	*assume* n (*number*(n))	Universal
18.	*consistent*(T)	Contradiction, L3, 1, 16
19.	*not*(*provable*($\forall y \neg A(\mathbf{k}, y), T$))	Contradiction, 18, 15
20.	*assume*($A(k, n)$)	Contradiction
21.	*gnp*($n, \forall y \neg A(\mathbf{k}, y), T$)	Forward, D3, 20
22.	*exists* n (*gnp*($n, \forall y \neg A(\mathbf{k}, y), T$))	Forward, 21
23.	*provable*($\forall y \neg A(\mathbf{k}, y), T$)	Forward, L2, 22
24.	*not*($A(k, n)$)	Contradiction, 19, 23, 20
25.	*provable*($\neg A(\mathbf{k}, \mathbf{n}), T$)	Reduction, L1, 1, 24
26.	*all* n (*number*(n) $\rightarrow$ *provable*($\neg A(\mathbf{k}, \mathbf{n}), T$))	Universal, 17, 25
27.	*not*(*provable*($\neg \forall y \neg A(\mathbf{k}, y), T$))	Reduction, D2, 1, 16, 26
28.	ω-*consistent*(T) $\rightarrow$ *not*(*provable*($\neg \forall y \neg A(\mathbf{k}, y), T$))	Implication, 16, 27
29.	*exists* F (*closed-formula*(F, T) & (*consistent*(T) $\rightarrow$ *not*(*provable*(F, T))) & (ω-*consistent*(T) $\rightarrow$ *not*(*provable*($\neg F, T$))))	Existential, 3, 15, 28
30.	*all* T (*fnt*(T) $\rightarrow$ *exists* F (...))	Universal, 1, 29

Roughly speaking, the composition heuristic constructs several closed formulas and the implication heuristic proves that one of these formulas is not provable if the theory is consistent and that its negation is not provable if the theory is ω-consistent. This formula is contained in steps 2 and 3 of

the proof. The proof of the first implication in step 15 is simply achieved by the contradiction and the forward heuristics. The proof of the second implication in step 28 requires a sophisticated interaction of the reduction, universal, contradiction, and forward heuristics.

Because the theorem to be proved is a universal proposition, the proof heuristic calls the universal heuristic. It introduces the assumption in step 1 which says that the antecedent of the theorem holds. Then, it calls the proof heuristic to prove the consequent of the theorem. Because this consequent is an existential proposition stating the existence of a closed formula F, the existential heuristic calls the composition heuristic. It constructs twenty-four closed formulas on the basis of the seven rules in Section 3. The existential heuristic first processes four short formulas. One of these formulas is in step 3. The numeral **k** in step 3 represents the Gödel number k of the formula in step 2. The existential heuristic successively substitutes the four formulas for the variable F in the scope of the existential proposition in the theorem. Then, it repeatedly calls the proof heuristic within a time limit which is one minute at the beginning and which is doubled in each repetition. This fails many times until the time limit is 128 minutes and the formula in step 3 is processed. In order to prove the first implication in the instantiated scope of the existential proposition, the implication heuristic generates step 4 which assumes that the antecedent of the implication holds. Then, it calls the proof heuristic to prove its consequent

$$not(provable(\forall y \neg A(num(k), y), T)). \tag{7}$$

Because (7) is the negation of a proposition, the negation heuristic calls the reduction heuristic within a time limit. Because this fails, it calls the contradiction heuristic which produces step 5 by applying Definition 1. Then, it generates step 6 containing the assumption that the proposition holds and calls the forward heuristic which produces steps 7–13 by applying definitions and lemmas to preceding proof steps. Because steps 12 and 13 yield a contradiction, the contradiction heuristic generates step 14 containing the negation (7) of the assumption in step 6. Finally, the implication heuristic generates the implication in step 15 from steps 4 and 14.

The proof of the second implication requires a sophisticated interaction of the reduction, universal, contradiction, and forward heuristics. The implication heuristic generates step 16 which assumes that the antecedent of the implication holds and calls the proof heuristic to prove its consequent

$$not(provable(\neg\forall y \neg A(num(k), y), T)). \tag{8}$$

Because (8) is a negation, the negation heuristic calls the reduction heuristic within a time limit. It reduces the goal to prove the negation (8) to the

subgoal to prove the universal proposition

$$all\,n(number(n) \rightarrow provable(\neg \mathrm{A}(num(k), num(n)), T)) \qquad (9)$$

as follows: It matches the consequent $not(provable(\neg \forall x F, T))$ in Definition 2 against the negation (8) which yields the substitution T/T, y/x, and $\neg \mathrm{A}(num(k), \mathrm{y})/F$. The application of this substitution to the antecedent

$$all\,n(number(n) \rightarrow provable(sub(F, x, num(n)), T)) \qquad (10)$$

in Definition 2 yields the universal proposition (9). In order to prove (9), the universal heuristic introduces the assumption in step 17 that the antecedent of (9) holds. By means of the implication

$$not(A(k, n)) \rightarrow provable(\neg \mathrm{A}(num(k), num(n)), T) \qquad (11)$$

in Lemma 1, the reduction heuristic reduces the goal to prove the consequent $provable(\neg \mathrm{A}(num(k), num(n)), T)$ of the universal proposition (9) to the subgoal to prove the negation $not(A(k, n))$. The negation heuristic calls the reduction heuristic within a time limit. Because this fails, it calls the contradiction heuristic which produces steps 18 and 19 by applying Lemma 3. Then, it generates the assumption in step 20 that the proposition $A(k, n)$ holds and calls the forward heuristic which produces steps 21–23 by applying Definition 3 and Lemma 2. Because steps 19 and 23 yield a contradiction, the contradiction heuristic generates step 24 containing the negation $not(A(k, n))$ of the assumption $A(k, n)$ in step 20. Now, the application of the implication (11) in Lemma 1 to this negation yields step 25. Then, the universal heuristic produces the proposition (9) in step 26 from steps 17 and 25. The application of Definition 2 to steps 16 and 26 yields the proposition (8) in step 27. The implication heuristic generates the implication in step 28 from steps 16 and 27. Now, the existential heuristic applies the rule of existence introduction to steps 3, 15, and 28 which yields the existential proposition in step 29, i.e., the existence of an undecidable formula. Finally, the universal heuristic generates Gödel's theorem in step 30 from steps 1 and 29.

Ordinarily, proofs of Gödel's theorem use Cantor's diagonal method to construct the undecidable formula $\forall \mathrm{y} \neg \mathrm{A}(\mathbf{k}, \mathrm{y})$, where k is the Gödel number of the formula $\forall \mathrm{y} \neg \mathrm{A}(\mathrm{x}, \mathrm{y})$ (see [11, pp. 6, 205, 207]). In SHUNYATA's proof, the undecidable formula is produced by the composition heuristic which applies the rules in Section 3. Since the composition heuristic and these rules are rather elementary and do not contain Cantor's method, it seems justified to say that SHUNYATA implicitly rediscovered Cantor's diagonal method while it was proving Gödel's theorem.

6. Discussion

The power of the SHUNYATA program is due to its heuristics and the language CL which are related to reasoning processes of mathematicians and guide the search for a proof (see Bledsoe [8]). In order to construct a formula, a mathematician may apply elementary rules for the formation of formulas such as those in Section 3. In order to construct a set, he may compose functions representing elementary concepts to form a predicate which determines the elements of a set. The composition heuristic in SHUNYATA simulates these reasoning processes. In the proof of Gödel's theorem, it produces an undecidable formula which is the central "idea" of the proof. In the proof of the Bolzano–Weierstrass theorem which says that a bounded infinite set S has an accumulation point (see Apostol [6, p. 43]), it produces a predicate $P(x)$ defined by

$$\mathit{finite}\{y \in S : y < x\} \tag{12}$$

which means that the set of all $y \in S$ with $y < x$ is finite. The proof that the supremum of the set $\{x : P(x)\}$ is an accumulation point of S is rather straightforward. The examples show that the composition heuristic can generate the central "ideas" of significant proofs in mathematical logic and analysis. Roughly speaking, the heuristic does not produce a combinatorial explosion for the following reasons: It first generates short formulas and predicates which are used for the construction of longer ones. Furthermore, subsequent heuristics applying these formulas and predicates such as the existential heuristic first process short formulas and predicates. Because the central "ideas" of many proofs have a rather short representation, a suitable formula or predicate is often found before many formulas or predicates are generated and processed. Another reason is that the formulas and predicates must satisfy constraints. In the proof of Gödel's theorem, the formula to be constructed must be closed. Furthermore, the composition heuristic first uses elementary concepts that are contained in the theorem to be proved. For example, the concept of finite sets in (12) is contained in the Bolzano–Weierstrass theorem because the theorem refers to an infinite set, i.e., a set that is not finite. Another source of SHUNYATA's power is the language CL which, for example, incorporates the concept of substitution into the special reasoning procedures. The ordinary notation $F(t)$ of the result of substituting a term t for the variable x in a formula $F(x)$ is represented by the expression $sub(F, x, t)$ (see Section 2). Formulas are normalized such that the formula F in the first argument of an expression $sub(F, x, t)$ does not contain propositional connectives or quantifiers. This normalization is used because some definitions and lemmas such as Lemma 5 do not require a substitution of a term t for the variable x in a formula $F(x)$, i.e., their

formalization does not contain an expression of the form $sub(F, x, t)$. For example, if the formula $\exists y A(\mathbf{k}, y)$ in step 11 in Table 1, which is represented by the normalized expression

$$\exists y\, sub(A(x, y), x, \mathbf{k}), \tag{13}$$

was replaced by the equivalent expression

$$sub(\exists y A(x, y), x, \mathbf{k}), \tag{14}$$

the formula $\exists x F$ in the antecedent of Lemma 5 could not be matched against (14) in order to generate step 12. But the formula $\exists x F$ in Lemma 5 can easily be matched against the normalized expression (13) which yields the substitution y/x and $sub(A(x, y), x, \mathbf{k})/F$. Thus, the proof of Gödel's theorem in Table 1 requires different equivalent representations for the result $F(t)$ of substituting a term t for the variable x in a formula $F(x)$. SHUNYATA contains heuristics that control the application of other heuristics. In this way, it automatically chooses rules of inference and controls their application without any human intervention. A simple example is the negation heuristic which controls the reduction and the contradiction heuristics by time limits. This entails that the proof of a negation can be achieved by SHUNYATA although its proof by a particular heuristic may fail because of a combinatorial explosion. SHUNYATA contains further powerful heuristics [3]. In order to prove the existence of a particular element, a heuristic derives properties of such an element and applies them to its definition. Another heuristic selects definitions and lemmas needed in a proof from a knowledge base of definitions and lemmas.

The composition heuristic, which produces compositions of functions, was originally developed as a learning procedure that analyzes simple proofs in group theory. For example, the analysis of a proof of the simple theorem that $(x^{-1})^{-1} = x$ holds for all elements x of a group produced a powerful theorem prover. The theorem prover, which is composed of elementary functions such as the subset relation, is constructed by applying rules giving the domains and ranges of these functions. This yields more and more complex compositions of these functions until the complete theorem prover is generated (see Ammon [2]). In an experiment, the theorem prover generated proofs of nine further theorems in group theory without any human intervention. One of the theorems, which says that $x^2 = 1$ implies group commutativity, is the "limit of the capability" of the heuristic theorem prover ADEPT developed at MIT in the mid-sixties (Loveland [12, p. 13]). ADEPT was developed manually. In contrast, SHUNYATA automatically developed the "ideas" for the theorem prover on the basis of elementary functions, implemented them in a program in the language

CL, and applied the resulting theorem prover to the new theorems without any human intervention. By means of a single additional heuristic, this theorem prover can even generate a proof of SAM's Lemma which is a theorem in modular lattice theory and a rather difficult problem for conventional theorem provers (see Antoniou and Ohlbach [5, p. 919]).[5] The learning procedure was also applied to the development of programs such as an infimum program from input–output pairs, the generation of mathematical conjectures such as Goldbach's conjecture from elementary arithmetical functions, the rediscovery of laws of physics such as Ohm's law on the basis of experimental data, and the construction of polygon concepts from line drawings (see Ammon and Stier [4]). The examples show that the learning procedure is rather general and powerful. Hence, an automatic development of SHUNYATA's heuristics might be achieved by the learning procedure. Because its input consists of elementary functions the programs in its output are composed of, this suggests that an "intelligent machine" can autonomously develop its programs and its heuristics in an experimental process which is based on a universal set of elementary functions.

7. Conclusions

The SHUNYATA program contains heuristics which are related to reasoning processes of mathematicians and guide the search for a proof. For example, the composition heuristic produces formulas and sets which form the central "ideas" of proofs. In the proof of Gödel's theorem, it produced an undecidable formula which is the most essential part of other forms of Gödel's theorem. Because this formula is ordinarily constructed by means of Cantor's diagonal method, SHUNYATA thus implicitly rediscovered Cantor's diagonal method while it was proving Gödel's theorem. Some heuristics control the application of other heuristics, for example, by time limits which interrupt a heuristic if it achieves no result. The proofs are represented in the language CL which incorporates concepts such as the concept of substitution into special reasoning procedures and which models the ordinary representation of proofs in mathematics textbooks. SHUNYATA produced readable proofs of significant theorems in mathematical logic and standard analysis without any human intervention, for example, proofs of Gödel's theorem and the Bolzano–Weierstrass theorem. Further experiments with a learning procedure suggest that an automatic construction of SHUNYATA's heuristics on the basis of a universal set of elementary functions is feasible.

[5]Ammon [2] describes an earlier comparable experiment which also applied the learning procedure to a simple proof. The proofs of SAM's Lemma generated in the two experiments are identical.

Future work should concentrate on the automatic discovery of new proofs of further significant theorems and the automatic construction of heuristics SHUNYATA is composed of.

Acknowledgment

The author would like to thank the reviewers for criticism and helpful suggestions. Thanks also to Andreas Keller and Alexander Reinefeld for providing useful comments on an earlier draft. This work was supported in part by the German Science Foundation (DFG).

References

[1] K. Ammon, Discovering a proof for the fixed point theorem: a case study, in: *Proceedings ECAI-88*, Munich, Germany (1988) 613–618.
[2] K. Ammon, The automatic acquisition of proof methods, in: *Proceedings AAAI-88*, St. Paul, MN (1988) 558–563.
[3] K. Ammon, Automatic proofs in mathematical logic and analysis, in: *Proceedings 11th International Conference on Automated Deduction*, Saratoga Springs, NY (1992) 4–19.
[4] K. Ammon and S. Stier, Constructing polygon concepts from line drawings, in: *Proceedings ECAI-88*, Munich, Germany (1988) 663–665.
[5] G. Antoniou and H.J. Ohlbach, Terminator, in: *Proceedings IJCAI-83*, Karlsruhe, Germany (1983) 916–919.
[6] T.M. Apostol, *Mathematical Analysis* (Addison-Wesley, Reading, MA, 1957).
[7] A.M. Ballantyne and W.W. Bledsoe, Automatic proofs of theorems in analysis using nonstandard techniques, *J. ACM* **24** (1977) 353–374.
[8] W.W. Bledsoe, Non-resolution theorem proving, *Artif. Intell.* **9** (1977) 1–35.
[9] G. Gentzen, Untersuchungen über das logische Schließen, *Math. Z.* **39** (1934) 176–210, 405–431.
[10] M. Kaufmann, DEFN-SK: an extension of the Boyer–Moore Theorem Prover to handle first-order quantifiers, Tech. Report 43, Computational Logic, Inc., Austin, TX (1989).
[11] S.C. Kleene, *Introduction to Metamathematics* (North-Holland, Amsterdam, 1952).
[12] D.W. Loveland, Automated theorem proving: a quarter century review, in: W.W. Bledsoe and D.W. Loveland, eds., *Automated Theorem Proving: After 25 Years* (American Mathematical Society, Providence, RI, 1984) 1–42.
[13] N. Shankar, Proof checking metamathematics, Tech. Report 9, Computational Logic, Inc., Austin, TX (1987).